Modeling J2EE Applications using Oracle9*i* Designer and Oracle9*i* JDeveloper

An Oracle White Paper
April 2002

ORACLE

Modeling J2EE Applications using Oracle9*i* Designer and Oracle9*i* JDeveloper

Modeling J2EE Applications using Oracle9*i* Designer and Oracle9*i* JDeveloper

EXECUTIVE OVERVIEW

Transactional applications development has always required the integration of the often-conflicting interests of data server design and application design. This does not change with three-tier component-based J2EE applications based on Business Components for Java (BC4J). This paper examines how Oracle9*i* Designer and Oracle9*i* JDeveloper can be used together to deliver an effective database applications development environment.

INTRODUCTION

Using today's modern development tools, there are a number of different approaches to professional application development:

- Hand-code
- Component-based architectures
- Declarative Rapid Application Development (RAD) tools
- Model-based development
- Custom Accelerators

APPLICATION DEVELOPMENT TODAY

Approaches

Hand-code

You can hand-code using 3GL-type languages such as Java and C++, most probably using a powerful Integrated Development Environment (IDE) that will contain coding tools such as profilers, debuggers, compilers, and deployment utilities. This approach is supremely flexible, but is costly in terms of resources and quality: lots of programmers writing lots of code will create lots of bugs, no matter how talented or conscientious they are.

This then forces development organizations to curtail the development lifecycle early so as to increase the number of testing and bug-fixing iterations; with perhaps as much as three to four months of each development cycle being taken up with

QA, code 'polishing', and bug-fixing. With the recent emphasis on accelerating time to market, and the associated decrease in the time between software releases, we see pressure being applied to the amount of time devoted to actually coding new features!

JDeveloper 3.2: Java & XML Coding

Oracle currently provides the technologically advanced JDeveloper IDE (version 3.2.3) to fulfill the Java coding environment need. Full Java language and end-to-end XML support, coupled with the capability to support a number of deployment and middle-tier architectures (JSP, Servlets, EJBs, CORBA Objects) and some significant productivity tools have made it the tool of choice for many Java software engineers.

Component-based architectures

An additional overhead from which the hand-coding approach suffers is the mismatched interfaces of today's object-oriented coding languages and the data persistency richness and robustness afforded by the modern relational database.

A powerful solution to this problem can be found by employing an architecture based on components to provide the 'glue' between the 3GL programs and the relational database. This technique of using ready-made components affords an increase in programmers' productivity while still maintaining a large degree of flexibility. When the component architecture is standardized, and developed and maintained as a software product itself, it becomes an extremely powerful tool in the struggle for productivity and quality. J2EE is just such an architecture.

Business Components For Java: Component Development

As ever though, the flexibility of a generic standard is always compromised by its very nature, and it is with that in mind that Oracle developed Business Components For Java (BC4J). These software components augment and integrate with J2EE components to provide a robust framework that supports the kind of rich relational database interaction that you would expect from the world's leading database supplier. Predictably, BC4J is an integral part of JDeveloper, the IDE providing a number of wizards that make the creation of the BC4J components extremely easy The BC4J framework however has been designed to be server-agnostic, and is completely open, thus providing a 'white box' approach, as opposed to the proprietary 'black box' approach.

Declarative Rapid Application Development (RAD) tools

RAD tools have been with us for many years now and provide us with the ability to define how a particular software object will behave at certain events and, using declarative sets of properties, define how it will look in the user interface (UI).

These tools radically increase productivity, the amount of code required being slashed. However, there is a trade off in terms of flexibility, as the underlying

enabling software will always restrict, to a greater or lesser extent, the ability of the software engineer to do whatever they want.

Forms 6i: Declarative

Oracle Forms 6*i* is a classic RAD tool. Evolved over many years, it has been used extensively by literally hundreds of thousands of developers worldwide, including notably Oracle Applications E-Business Suite, to develop richly interactive applications that are tightly integrated with the relational database.

Model-based development

Developing application systems using graphical models, and then generating significant quantities of code from these models, has been a technique available to application developers for many years. The variety and scope of these models, and the quantity, quality, and completeness of the generated code varies from tool to tool, but the result is always the same; productivity is increased.

However, wherever code generation is employed, the engineers who wrote the generators will have had to analyze the requirements and create generic set-piece structures that can then be fleshed out with the context at generation time. No matter how complete the analysis or the capability of the generator, you will always find something more appropriate for a hand-coded approach. In the vast majority of cases, this is offset by the huge gains in productivity and quality.

Designer 6i: Modeling & Generation

Oracle Designer has been Oracle's application development modeling tool since the late '80s and supports a number of modeling techniques that have been developed over the last 15 to 20 years. From systems analysis techniques such as *entity relationship modeling* and *process modeling*, to *server design, generation, and capture*, and program generators such as the *Forms Generator* and *Reports Generator*, Designer has a proven track record as one of the world's leading modeling and generation tools.

Custom Accelerators

To offset the issue of gains in productivity leading to loss of flexibility, and of course *vice versa*, there is the possibility of extending the model-based and declarative solutions with generic framework code, templates, and patterns that can be customized during design and build. This provides in effect a kick-start to a project so that much of the architectural groundwork has already been done. You get the benefit of productivity enhancing modeling and generation while regaining the flexibility of component- or pattern-based architectures.

Oracle JHeadstart

Oracle Consulting has long been involved in offering framework services to customers, JHeadstart being the latest of these. Among other features, the JHeadstart service provides a mechanism to take existing Designer artifacts and

generate BC4J components from them. This is an extremely powerful capability, and we will cover it in more detail later in the paper.

Figure 1 summarizes the issues discussed above, and illustrates how custom accelerators can claw back some of the flexibility relinquished in the search for productivity

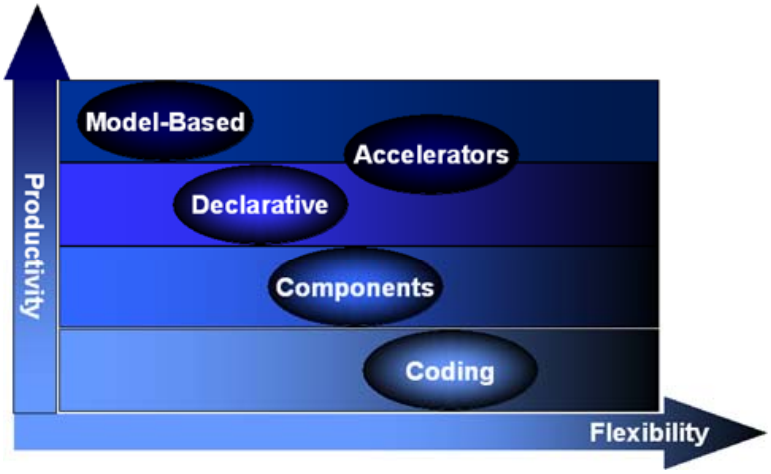

Figure 1: Application Development Approaches

Configuration Management

One of the common requirements of application development in any environment is the need to support team working and versioning.

With the 6*i* release of Designer, Oracle introduced file support, versioning (check in and check out), and configuration management (branching, merging, release) into the Designer repository. This technology has been extended to support Oracle's other developments tools, most notably Forms 6*i* and JDeveloper 3.2.

This has led to the evolution of a new tool, Oracle Software Configuration Manager (SCM).

HOW WILL ORACLE9*i* HELP?

What is Oracle9*i*?

Oracle9*i* is an integrated set of products that provides the complete solution to the application developer:

- *Oracle9i – the database*: the most powerful and robust object-relational database in the world today
- *Oracle9i Application Server – Oracle9iAS*: a complete and integrated set of components, tools, and servers to enable the deployment of J2EE application, multi-tier portal applications, and the rich Java UI of Oracle Forms
- *Oracle9i Developer Suite – Oracle9iDS*: the tools that support the database and create the applications that run on the servers

The main goals of the Oracl9*i* platform are to provide software products that are:

- Complete
- Integrated
- Standard

What is Oracle9iDS?

The components of Oracle9iDS can be viewed in two main functional areas:

- Rapid Application Development tools (RAD):
  - Oracle9*i* Software Configuration Manager (SCM)
  - Oracle9*i* Designer
  - Oracle9*i* Forms Developer
  - Oracle9*i* JDeveloper and BC4J
  - Developer's Kits
- Business Intelligence tools (BI):
  - Oracle9*i* Reports Developer
  - Oracle9*i* Discoverer Administrator
  - Oracle9*i* Warehouse Builder
  - Oracle9*i* Clickstream Intelligence Builder

This paper is concerned with Oracle9*i* Designer and Oracle9*i* JDeveloper.

Oracle9*i* Designer

The Oracle9iDS release of Oracle9*i* Designer is a consolidation of the existing Designer 6i against the new software stack. Oracle9*i* Designer will design, generate, capture, and install into an Oracle9*i* database. It will co-exist in the same Oracle Home as the other 9*i* products, and generate and capture the sister tools in the suite: Oracle9*i* Forms and Oracle9*i* Reports. Generated Web PL/SQL applications will run on a 9*i*AS server.

Oracle9*i* JDeveloper

The Oracle9iDS release of JDeveloper has undergone some significantly more fundamental improvements, and, as illustrated in figure 2, now provides an end-to-end tool for developing J2EE applications in an enterprise environment.

Figure 2: Oracle9*i* JDeveloper at a glance

The entire IDE has been recoded in Java, and the tool now boasts full support for the J2EE environment and full integration with the Oracle9*i* AS Java solution Oracle9*i*AS Containers for J2EE (OC4J).

For the first time the IDE has UML Modeling components. These support Class modeling, Activity modeling, Javacode generation and synchronization, and Enterprise Application Integration (EAI) with the generation of Oracle AQs and Workflow.

For the Java programmer there are new profiling, editing, and debugging tools, and a tool called CodeCoach that makes suggestions for coding improvements. There is also richer support for Java Server Pages (JSP), BC4J, and XML.

Database integration has also been enhanced and, in addition to the existing configuration management support for Oracle SCM, there is now support for Rational ClearCase and Gnu CVS.

UML Modeling

Oracle9*i* JDeveloper supports two types of modeling technique:

- Class Modeling
- Activity Modeling

Class Modeler

The Class Modeler supports all the UML notation that you would expect:

- Java Classes
- Interfaces
- Domains
- Packages
- Associations (directional and non-directional, strong and weak aggregation)
- Generalization
- Realization
- Dependency

It also supports the modeling of BC4J Entity Objects (EOs). These objects map directly to relational tables in the database, and are one of the fundamental classes you need to create when developing a BC4J application. You can either create EOs in a diagram then generate the resulting tables to the database, or more typically, capture the structure of the table from the database and create the EO in a diagram. This process will also capture key constraint information, with foreign key constraints resulting in EO associations in the resulting model.

Code Generation and Synchronization

Possibly the most powerful feature of the Class Modeler is its ability to generate Java from a model, and then maintain synchronization between the code and the model from then on. In fact you don't have to start with a model. You can create the model from the code. This is sometimes known as *reverse engineering* the model.

The synchronization is two-way:

- When you change the model, the code is updated
- When you change the code, the model is updated

With the BC4J EO model, the Class Modeler also generates an XML meta data model, and this too is kept synchronized with the model.

Activity Modeler

The Activity Modeler models e-business process and objects states, and supports the following UML elements and structures:

- Activities
- Object states
- Transitions
- Pseudostates (initial, final, forks, joins)
- Swimlanes

In addition to this generic activity modeling capability, the Activity Modeler has been extended to support and generate Enterprise Application Integration definitions that support Oracle AQ and Oracle Workflow.

XML, SCM, and XMI

Both the Class Modeler and Activity Modeler store their artifacts as XML definitions; there being one file for each object in a model (Java Class, Objects State, and so on), and one for each diagram.

Oracle9*i* JDeveloper supports three major configuration management products:

- Oracle SCM
- Rational ClearCase
- Gnu CVS (`http://www.gnu.org/software/cvs/cvs.html`)

Because of this full integration, the entire IDE, including the modeling tools, supports team working.

Oracle9*i* JDeveloper currently supports import from the following UML modeling tools using XMI standards:

- Rational Rose
- TogetherSoft TogetherJ

If you have been developing class or activity models using either of the above tools, then it is fairly straightforward to migrate them to JDeveloper to take advantage of the IDE and code generation and synchronization features.

ORACLE9*i*DS: A NEW MODELING APPROACH

Overview

For the first time Oracle has products that support the traditional Information Engineering approach to modeling, as well as the latest UML approach, in an integrated package: Oracle9*i*DS.

People will need to use Designer to develop and maintain their relational database server objects, the persistence medium of choice for the majority of data-oriented J2EE application. They will also need to use JDeveloper to model, generate and code for J2EE, using BC4J to enable the object-relational mapping to the server objects.

Use Cases

This section of the paper contains a number of use cases that outline how Oracle9*i* Designer and Oracle9*i* JDeveloper might be used 'in concert' as it were to support a number of different application development problems.

1. **I have a Designer server model on which I want to build BC4J**

   The database design is being developed in Designer and the decision has recently been made to opt for a J2EE application architecture.

   - Iteratively generate the database design to support the application. This may consist of any of the relational objects and code currently supported by Oracle9*i* Designer, including PL/SQL objects (packages, procedures, functions), object types, database triggers, and so on.
   -
   - Capture the data structures into JDeveloper as BC4J EOs. You will be able to capture tables, database views, synonyms, materialized views, columns, primary keys, foreign keys (see figure 3 below).

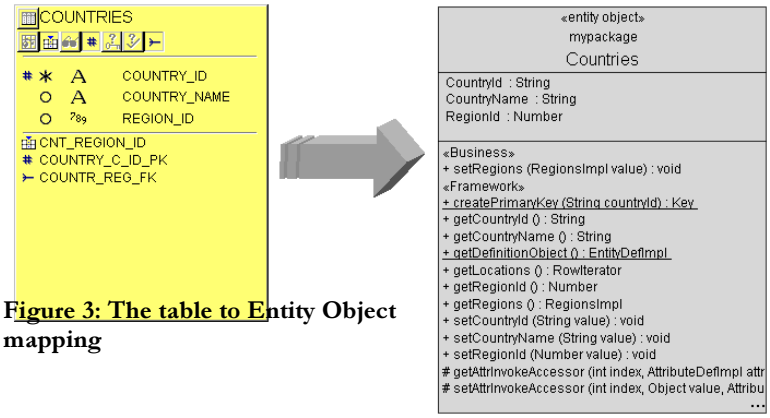

Figure 3: The table to Entity Object mapping

- This process also creates an XML file and Java class for each EO, and an XML file for each EO association.
- Then generate the remaining BC4J components (View Objects and Application Modules) using the Business Components generator.
  This process creates a View Object (VO) for each EO (consisting of an XML file and a Java class) and one Application Module (AM) per package (also consisting of an XML file and a Java class).

This use case could also apply to a complete system that has been developed over a number of years, perhaps using Forms as the UI. The decision has been taken to migrate the current architecture to a J2EE implementation, while leaving the data model relatively untouched.

2. I have a database that I want to visualize and refine before I start my BC4J development

The database is a legacy system that may even be in a non-Oracle database.

- Create an application system in Designer and capture the database objects into it.
  You will be able to capture virtually everything from the legacy server that you need to.
- Visualize the database design on a server model in Designer.
- Refine the design using Designer server modeling, and then generate it into a new database, or indeed the existing one.
  If you select the latter option, the Server Generator will analyze the existing database objects, identify the differences, product a report, and generate ALTER TABLE statements rather than dropping the table and recreating it.
- Create the BC4J components as detailed in use case #1 above.

3. I have been developing in JDeveloper and I want to verify my database design

Both JDeveloper 3 and Oracle9*i* JDeveloper allow you to create database objects from your EO definitions. This is an extremely useful capability, although limited; you cannot create much more than tables, columns, and keys.

- Generate the database from the EO definitions in JDeveloper.
- Capture the table, column and key definitions into Designer as detailed in use case #2 above.
- In Designer, visualize, verify and if necessary modify the table design.
- Add the required physical design objects, such as indexes, sequences, views, and DBA objects such as tablespaces, users, grants, and so on.
- Generate the completed database objects.
- Refresh the original EO definitions in JDeveloper:
    - Edit the EO in the diagram
    - Navigate to the Attributes tab and click the *New from Table*…or the *Synchronize* button.
    - Select any new columns that you added in Designer and generated into the database
      Pressing OK will update the existing EO XML and Java class files.
- Modify the VO and AM objects to reflect the use they may make of any new columns.

4. I am working on an EAI project and I want to manage dependencies on my Oracle Advanced Queues

The Activity modeler in Oracle9*i* JDeveloper can create Oracle Advanced Queues as part of an EAI solution. These are generated directly into the database and you might wish to manage them in Designer so that you can make use of Oracle SCM's sophisticated Dependency Analysis capabilities.

- Generate the AQs from JDeveloper
- Capture the AQs and their associated queue tables into Designer.

5. **I have been developing my UML models in another tool and I want to generate a robust persistency model**

   Oracle9*i* JDeveloper supports the import of class model components from another XMI compliant tool. If you have created class models in either Rational Rose or TogetherSoft TogetherJ then you will be able to migrate the UML and XMI compliant artifacts from the source tool to JDeveloper Class Modeler.

   - Generate an XML export file containing the XMI data from the source tool.
   - Create an empty project in JDeveloper.
   - Create a new UML diagram in the empty project.
   - Select *Class Diagrams from XMI Import* then click OK.
   - Select the XML file containing the XMI that has been exported from your other modeling tool and click Open.
     One class diagram is created for each package in the model, and one diagram is created for all the other elements that are not in a package, for example, those at project level.
     Relationships, such as associations, generalizations and realizations, between packages in the model will not be drawn on the diagrams.

6. **I want to use a single SCM system for my Designer and JDeveloper artifacts**

   As soon as a development team grows beyond a few engineers, it is likely that you will want to pursue a configuration management solution that prevents files that one person is working on being compromised by another. Oracle9*i* JDeveloper now supports three SCM systems, and there are plans to support more in the future. Designer only supports Oracle SCM.

   - In Designer, enable versioning using the Repository Administration Utility (RAU):
     - **Options > Enable Version Support …**
   - In JDeveloper, enable source control from the File menu:
     - **File > Source Control > Enable**
     - Create a new connection in the Oracle9*i* SCM node in the JDeveloper navigator.

7. **I want to leverage my Designer module definitions in the J2EE world**

   Designer has rich schema and program module meta-models from which you can generate complete applications. These meta-models share semantic similarities with the BC4J model. Oracle Consulting have been developing JHeadstart, which incorporates Project Atlantis, and is part of their suite of Custom Accelerators. This technology consists of generators to pull information from the Designer schema and module elements, and then create Java and XML based applications within the BC4J framework in JDeveloper. The model is 'pull' rather than 'push'.

   Appendix 1 contains an overview of the JHeadStart offering.

Sharing Repository Definitions

Given the previous use cases, a rich Designer model (from which you may already have been generating complex Forms applications), and the flexibility of JHeadStart, it is conceivable that a J2EE application development project could not only *make use* of the Designer table and module definitions, but could *share* them with an ongoing Forms generation development.

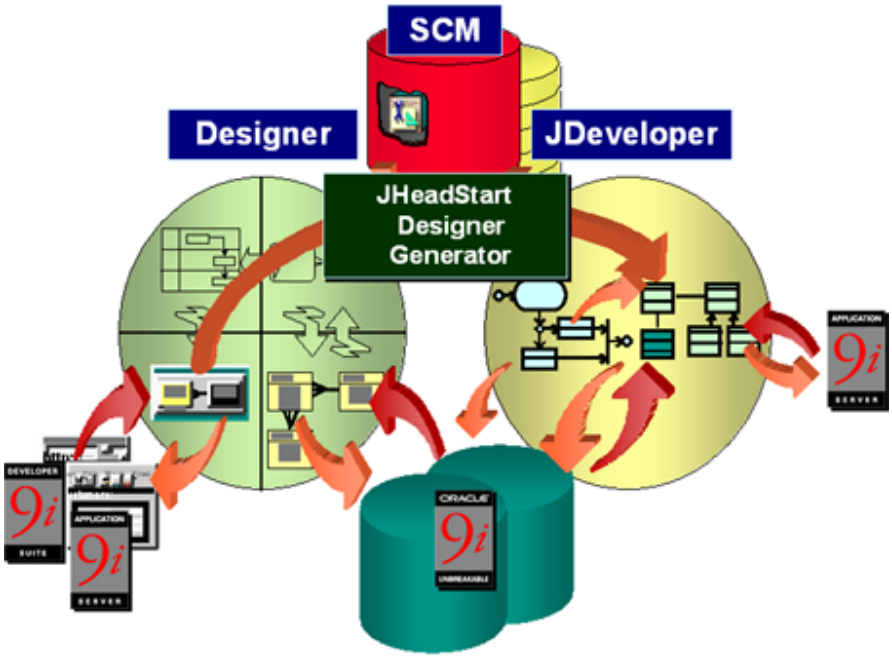

Figure 4: Developing Forms and J2EE applications from a common model

WHERE DO WE GO FROM HERE?

Oracle9*i* Designer: Release 2

With the Oracle9*i* stack stabilized in the first release of Oracle9*i* Designer, we plan to create some new features in Oracle9iDS Release 2. There will be better support for existing database objects, and further support for new Oracle9*i* database objects and structures.

Oracle9*i* JDeveloper: Release 2

In the same timeframe, we are intending to make the UML support both broader and deeper:

- More modeling techniques
- Wider XMI support
- UML Profiles
- Richer BC4J modeling

Designer to JDeveloper Migration Utilities

Using Oracle Consulting's JHeadStart service (see Appendix) as a starting point, Oracle are intending to develop an extensible transformer framework to enable the development of, not just schema and module transformers, but other transformers as well.

It is intended that this framework will also support custom transformers that could be developed by partners.

CONCLUSION

Oracle9*i* Development Suite contains both Designer and JDeveloper. Use them together to create J2EE applications.

Use Oracle9*i* Designer to:

- Model, generate and capture server objects
- Model module definitions

Use Oracle9*i* JDeveloper UML models to model, generate and capture:

- Class
- BC4J
- AQ

Use JHeadstart to generate Java / HTML applications from Designer module definitions.

APPENDIX: WHAT IS ORACLE JHEADSTART

Introduction

Oracle JHeadstart is Oracle Consulting's RAD approach for building J2EE applications. It enables fast, reliable, and repeatable development of complex transactional systems. It combines frameworks to implement the J2EE Model View Controller (MVC) architecture. It includes the JHeadstart Application Generator that allows you to specify your application definition declaratively in an XML file and enables you to generate into the frameworks.

What are the benefits of JHeadstart?

Oracle JHeadstart offers the following benefits:

- Fast and Productive Application Development
- Easy to learn
- Applications are easy to extend and customize
- Leverages proven frameworks
- Built on a flexible MVC architecture
- Delivers attractive user interface

Moving to Java with Oracle Consulting

Oracle Consulting can help you move to Java/J2EE and JHeadstart. There are a number of options:

Java/HTML Generation from Oracle Designer

If you use Oracle Designer the simplest way to move to Java is to generate Java/HTML applications directly from Oracle Designer. This allows you to generate Forms applications for your power users and HTML self-service applications for others from the same Repository. It means you do not have to start up a large upskilling effort or change your way of working. Oracle consulting offers a *JHeadstart Pilot* to demonstrate the possibilities of the JHeadstart-Oracle Designer approach for your organization.

Forms migration to Java/HTML

This option applies when you have already decided to go Java/J2EE and you are thinking of migrating your Oracle Forms to Java. Oracle Consulting offers a complete path from a *JHeadstart Migration Quick Scan* through a *Migration Pilot* and the actual *Migratio*n. The JHeadstart technology will reuse the meta data in the Oracle Designer Repository. The meta data can either be already available in the Repository or can be obtained by reverse engineering your Oracle Forms. In either case your investment in your current application will be protected as much as possible. When you use server side business logic, like CDM RuleFrame, in your

current application development approach, your business logic will be used by the Java application as well, further protecting your investment.

What Does JHeadstart Support?

JHeadstart Designer Generator

The JHeadstart Designer Generator uses meta data in the Oracle Designer Repository to drive the JHeadstart Application Generator. Elements that are used:

- Tables
- Views
- Columns
- Key Constraints
- Domains
- Module Components
- Table Usages
- LOVs

JHeadstart Application Generator

The JHeadstart Generator supports:

- *Layout Style*s: Form, Table, Table ranges, Table-Form, List, Find, Master-Detail
- *Widget*s: Date picker, text editor, check box, radio group, pop list, look ups, list of values,
- *Action*: multi-row insert, update and delete, sort on table headers
- *Navigation*: tab based navigation structure, button based navigation
- *Security*: Role based, log in facility
- *Error handlin*g: multiple error display with hyperlinks to field causing error

Modeling J2EE Applications using Oracle9*i* Designer and Oracle9*i* JDeveloper
April 2002
Author: Simon W Day
Contributing Authors: Steven Davelaar

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com